Part I: Introduction

1 Why Study Automata Theory?

2 Languages and Strings

1) Consider the language $L = \{1^n2^n : n > 0\}$. Is the string `122` in L?

   No. Every string in L must have the same number of `1`'s as `2`'s.

2) Let $L_1 = \{a^nb^n : n > 0\}$. Let $L_2 = \{c^n : n > 0\}$. For each of the following strings, state whether or not it is an element of L_1L_2:

a)	ε.	No.
b)	`aabbcc`.	Yes.
c)	`abbcc`.	No.
d)	`aabbcccc`.	Yes.

3) Let L_1 = {`peach`, `apple`, `cherry`} and L_2 = {`pie`, `cobbler`, ε}. List the elements of L_1L_2 in lexicographic order.

   `apple`, `peach`, `cherry`, `applepie`, `peachpie`, `cherrypie`, `applecobbler`, `peachcobbler`, `cherrycobbler` (We list the items shortest first. Within a given length, we list the items alphabetically.)

4) Let $L = \{w \in \{a, b\}^* : |w| \equiv_3 0\}$. List the first six elements in a lexicographic enumeration of L.

   ε, `aaa`, `aab`, `aba`, `abb`, `baa`

5) Consider the language L of all strings drawn from the alphabet {`a`, `b`} with at least two different substrings of length 2.
   a) Describe L by writing a sentence of the form $L = \{w \in \Sigma^* : P(w)\}$, where Σ is a set of symbols and P is a first-order logic formula. You may use the function $|s|$ to return the length of s. You may use all the standard relational symbols (e.g., =, ≠, <, etc.), plus the predicate *Substr*(s, t), which is *True* iff s is a substring of t.

      $L = \{w \in \{a, b\}^* : \exists x, y\ (x \neq y \wedge |x| = 2 \wedge |y| = 2 \wedge Substr(x, w) \wedge Substr(y, w))\}$.

   b) List the first six elements of a lexicographic enumeration of L.

      `aab`, `aba`, `abb`, `baa`, `bab`, `bba`

6) For each of the following languages L, give a simple English description. Show two strings that are in L and two that are not (unless there are fewer than two strings in L or two not in L, in which case show as many as possible).
   a) $L = \{w \in \{a, b\}^*$: exactly one prefix of w ends in `a`$\}$.

      L is the set of strings composed of zero or more `b`'s and a single `a`. `a`, `bba` and `bbab` are in L. `bbb` and `aaa` are not.

   b) $L = \{w \in \{a, b\}^*$: all prefixes of w end in `a`$\}$.

      $L = \varnothing$, since ε is a prefix of every string and it doesn't end in `a`. So all strings are not in L, including `a` and `aa`.

c) $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : \exists x \in \{\mathtt{a}, \mathtt{b}\}^+ (w = \mathtt{a}x\mathtt{a})\}$.

L is the set of strings over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ whose length is at least 3 and that start and end with a. aba, and aaa are in L. ε, a, ab and aa are not.

7) Are the following sets closed under the following operations? If not, what are their respective closures?

a) The language $\{\mathtt{a}, \mathtt{b}\}$ under concatenation.

Not closed. $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : |w| > 0\}$

b) The odd length strings over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ under Kleene star.

Not closed because, if two odd length strings are concatenated, the result is of even length. The closure is the set of all nonempty strings drawn from the alphabet $\{\mathtt{a}, \mathtt{b}\}$.

c) $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*\}$ under reverse.

Closed. L includes all strings of a's and b's, so, since reverse must also generate strings of a's and b's, any resulting string must have been in the original set.

d) $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ starts with $\mathtt{a}\}$ under reverse.

Not closed. L includes strings that end in b. When such strings are reversed, they start with b, so they are not in L. But, when any string in L is reversed, it ends in a. So the closure is $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ starts with $\mathtt{a}\} \cup \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ ends with $\mathtt{a}\}$.

e) $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ ends in $\mathtt{a}\}$ under concatenation.

Closed.

8) For each of the following statements, state whether it is *True* or *False*. Prove your answer.

a) $\forall L_1, L_2\ (L_1 = L_2 \text{ iff } L_1^* = L_2^*)$.

False. Counterexample: $L_1 = \{\mathtt{a}\}$. $L_2 = \{\mathtt{a}\}^*$. But $L_1^* = L_2^* = \{\mathtt{a}\}^* \neq \{\mathtt{a}\}$.

b) $(\varnothing \cup \varnothing^*) \cap (\neg\varnothing - (\varnothing\varnothing^*)) = \varnothing$ (where $\neg\varnothing$ is the complement of $\varnothing$).

False. The left hand side equals $\{\varepsilon\}$, which is not equal to $\varnothing$.

c) Every infinite language is the complement of a finite language.

False. Counterexample: Given some nonempty alphabet Σ, the set of all even length strings is an infinite language. Its complement is the set of all odd length strings, which is also infinite.

d) $\forall L\ ((L^R)^R = L)$.

True.

e) $\forall L_1, L_2\ ((L_1 L_2)^* = L_1^* L_2^*)$.

False. Counterexample: $L_1 = \{\mathtt{a}\}$. $L_2 = \{\mathtt{b}\}$. $(L_1 L_2)^* = (\mathtt{ab})^*$. $L_1^* L_2^* = \mathtt{a}^*\mathtt{b}^*$.

f) $\forall L_1, L_2 ((L_1{*}L_2{*}L_1{*})^* = (L_2 \cup L_1)^*)$.

True.

g) $\forall L_1, L_2 ((L_1 \cup L_2)^* = L_1^* \cup L_2^*)$.

False. Counterexample: $L_1 = \{\mathsf{a}\}$. $L_2 = \{\mathsf{b}\}$. $(L_1 \cup L_2)^* = (\mathsf{a} \cup \mathsf{b})^*$. $L_1^* \cup L_2^* = \mathsf{a}^* \cup \mathsf{b}^*$.

h) $\forall L_1, L_2, L_3 ((L_1 \cup L_2)\, L_3 = (L_1 L_3) \cup (L_2 L_3))$.

True.

i) $\forall L_1, L_2, L_3 ((L_1 L_2) \cup L_3 = (L_1 \cup L_3)\, (L_2 \cup L_3))$.

False. Counterexample: $L_1 = \{\mathsf{a}\}$. $L_2 = \{\mathsf{b}\}$. $L_3 = \{\mathsf{c}\}$.

$$\begin{aligned} (L_1 L_2) \cup L_3 &= \{\mathsf{ab}, \mathsf{c}\}. \\ (L_1 \cup L_3)\, (L_2 \cup L_3) &= (\mathsf{a} \cup \mathsf{c})(\mathsf{b} \cup \mathsf{c}) \\ &= \{\mathsf{ab}, \mathsf{ac}, \mathsf{cb}, \mathsf{cc}\} \end{aligned}$$

j) $\forall L ((L^+)^* = L^*)$.

True.

k) $\forall L (\varnothing L^* = \{\varepsilon\})$.

False. For any L, and thus for any L^*, $\varnothing L = \varnothing$.

l) $\forall L (\varnothing \cup L^+ = L^*)$.

False. $\varnothing \cup L^+ = L^+$, but it is not true that $L^+ = L^*$ unless L includes ε.

m) $\forall L_1, L_2 ((L_1 \cup L_2)^* = (L_2 \cup L_1)^*)$.

True.

3 The Big Picture: A Language Hierarchy

1) Consider the following problem: Given a digital circuit C, does C output 1 on all inputs? Describe this problem as a language to be decided.

   L = {<C> : C is a digital circuit that outputs 1 on all inputs}. <C> is a string encoding of a circuit C.

2) Using the technique we used in Example 3.8 to describe addition, describe square root as a language recognition problem.

   SQUARE-ROOT = {w of the form : <$integer_1$>, <$integer_2$>, where $integer_2$ is the square root of $integer_1$}.

3) Consider the problem of encrypting a password, given an encryption key. Formulate this problem as a language recognition problem.

   L = {x; y; z : x is a string, y is the string encoding of an encryption key, and z is the string that results from encrypting x using y}.

4) Consider the optical character recognition (OCR) problem: Given an array of black and white pixels, and a set of characters, determine which character best matches the pixel array. Formulate this problem as a language recognition problem.

   L = {<A, *C-list*, c> : A is an array of pixels, *C-list* is a list of characters, and c is an element of *C-list* with the property that A is a closer match to c than it is to any other element of *C-list*}.

5) Consider the language $A^nB^nC^n$ = {$a^nb^nc^n$: $n \geq 0$}, discussed in Section 3.3.3. We might consider the following design for a PDA to accept $A^nB^nC^n$: As each a is read, push two a's onto the stack. Then pop one a for each b and one a for each c. If the input and the stack come out even, accept. Otherwise reject. Why doesn't this work?

   This PDA will accept all strings in $A^nB^nC^n$. But it will accept others as well. For example, aabccc.

6) Define a PDA-2 to be a PDA with two stacks (instead of one). Assume that the stacks can be manipulated independently and that the machine accepts iff it is in an accepting state and both stacks are empty when it runs out of input. Describe the operation of a PDA-2 that accepts $A^nB^nC^n$ = {$a^nb^nc^n$: $n \geq 0$}. (Note: we will see, in Section 17.5.2, that the PDA-2 is equivalent to the Turing machine in the sense that any language that can be accepted by one can be accepted by the other.)

   M will have three states. In the first, it has seen only a's. In the second, it has seen zero or more a's, followed by one or more b's. In the third, it has seen zero or more a's, one or more b's, and one or more c's. In state 1, each time it sees an a, it will push it onto both stacks. In state 2, it will pop one a for each b it sees. In state 3, it will pop one a for each c it sees. It will accept if both stacks are empty when it runs out of input.

4 Some Important Ideas Before We Start

1) Describe in clear English or pseudocode a decision procedure to answer the question: given a list of integers *N* and an individual integer *n*, is there any element of *N* that is a factor of *n*?

```
decidefactor(n: integer, N: list of integers) =
    For each element i of N do:
        If i is a factor of n, then halt and return True.
    Halt and return False.
```

2) Given a Java program *p* and the input 0, consider the question, "Does *p* ever output anything?"
   a) Describe a semidecision procedure that answers this question.

   Run *p* on 0. If it ever outputs something, halt and return *True*.

   b) Is there an obvious way to turn your answer to part (a) into a decision procedure?

   No and there isn't any other way to create a decision procedure either.

3) Let $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*: w = w^R\}$. What is *chop*(*L*)?

chop(*L*) = $\{w \in \{\mathtt{a}, \mathtt{b}\}^*: w = w^R$ and $|w|$ is even$\}$.

4) Are the following sets closed under the following operations? Prove your answer. If a set is not closed under the operation, what is its closure under the operation?

   a) $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ ends in $\mathtt{a}\}$ under the function *odds*, defined on strings as follows: *odds*(*s*) = the string that is formed by concatenating together all of the odd numbered characters of *s*. (Start numbering the characters at 1.) For example, *odds*(`ababbbb`) = `aabb`.

   Not closed. If $|w|$ is even, then the last character of *odds*(*w*) will be the next to the last character of *w*, which can be either `a` or `b`. For any *w*, $|odds(w)| \leq |w|$, and the shortest string in *L* has length 1. So the closure is $\{\mathtt{a}, \mathtt{b}\}^+$.

   b) FIN (the set of finite languages) under the function *oddsL*, defined on languages as follows:
   $oddsL(L) = \{w : \exists x \in L\ (w = odds(x))\}$

   FIN is closed under the function *OddsL*. Each string in *L* can contribute at most one string to *OddsL*(*L*). So $|OddsL(L)| \leq |L|$.

   c) INF (the set of infinite languages) under the function *oddsL*.

   INF is closed under the function *OddsL*. If INF were not closed under *OddsL*, then there would be some language *L* such that $|L|$ is infinite but $|OddsL(L)|$ were finite. If $|OddsL(L)|$ is finite, then there is a longest string in it. Let the length of one such longest string be *n*. If $|L|$ is infinite, then there is no longest string in *L*. So there must be some string *w* in *L* of length at least $2n + 2$. That string *w* will cause a string of length at least $n + 1$ to be in *OddsL*(*L*). But if $|OddsL(L)|$ is finite, the length of its longest string is *n*. Contradiction.

   d) FIN under the function *maxstring*, defined in Example 8.22**Error! Reference source not found.**.

   FIN is closed under the function *maxstring*. Each string in *L* can contribute at most one string to *maxstring*(*L*). So $|maxstring(L)| \leq |L|$.

e) INF under the function *maxstring*.

> INF is not closed under *maxstring*. `a`* is infinite, but *maxstring*(`a`*) = ∅, which is finite.

5) Let Σ = {`a`, `b`, `c`}. Let *S* be the set of all languages over Σ. Let *f* be a binary function defined as follows:

$$f: S \times S \rightarrow S$$
$$f(x, y) = x - y$$

Answer each of the following questions and defend your answers:

a) Is *f* one-to-one?

> No, *f* is not one-to-one. Counterexample: {`a`, `b`, `c`} - {`b`, `c`} = {`a`} and {`a`, `b`} - {`b`} = {`a`}.

b) Is *f* onto?

> Yes, *f* is onto. For any language *L*, *L* - ∅ = *L*.

c) Is *f* commutative?

> No. *f* is not commutative. Counterexample: {`a`, `b`} - {`b`} = {`a`}, but {`b`} - {`a`, `b`} = ∅.

6) Describe a program, using *choose*, to:

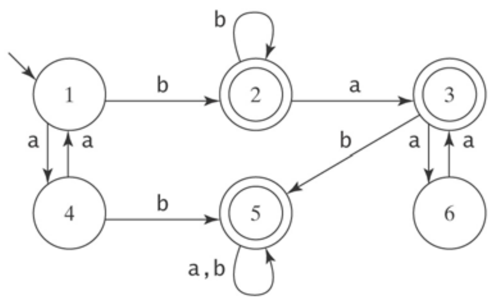
a) Play Sudoku 💻.

b) Solve Rubik's Cube® 💻.

Part II: Regular Languages

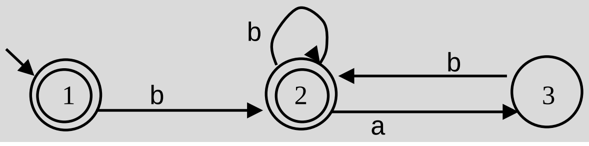
5 Finite State Machines

1) Give a clear English description of the language accepted by the following FSM:

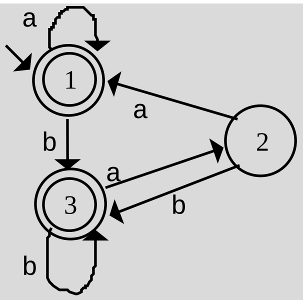


All strings of a's and b's consisting of an even number of a's, followed by at least one b, followed by zero or an odd number of a's.

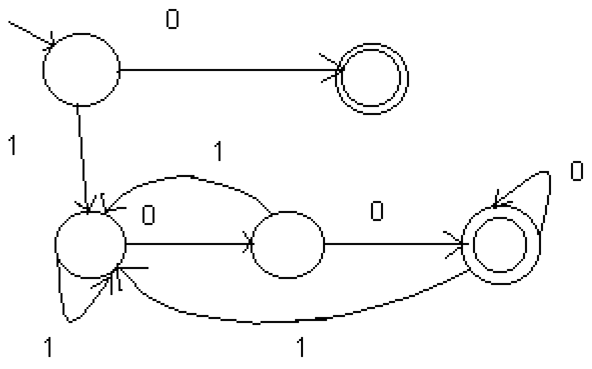
2) Build a deterministic FSM for each of the following languages:
   a) {*w* ∈ {a, b}* : every a in *w* is immediately preceded and followed by b}.



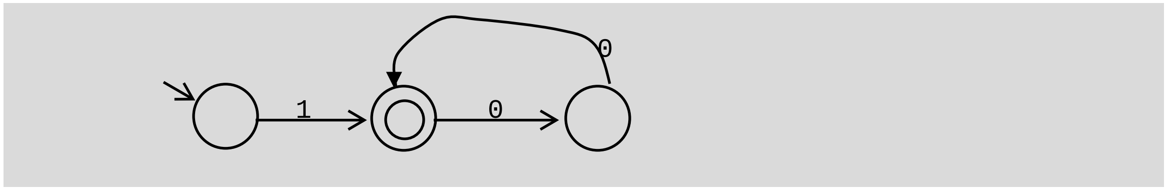
   b) {*w* ∈ {a, b}* : *w* does not end in ba}.



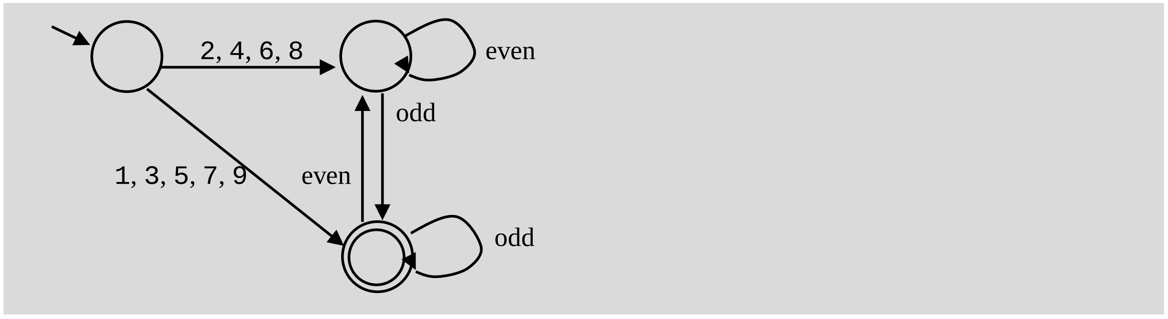
c) {*w* ∈ {0, 1}* : *w* corresponds to the binary encoding, without leading 0's, of natural numbers that are evenly divisible by 4}.



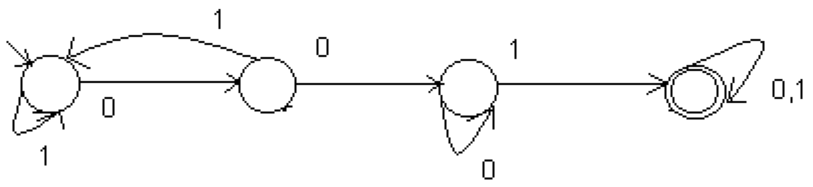
d) {*w* ∈ {0, 1}* : *w* corresponds to the binary encoding, without leading 0's, of natural numbers that are powers of 4}.



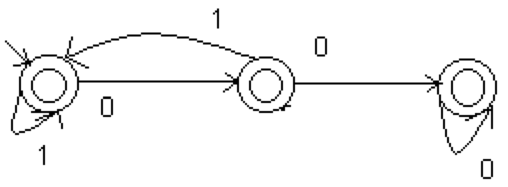
e) {*w* ∈ {0-9}* : *w* corresponds to the decimal encoding, without leading 0's, of an odd natural number}.



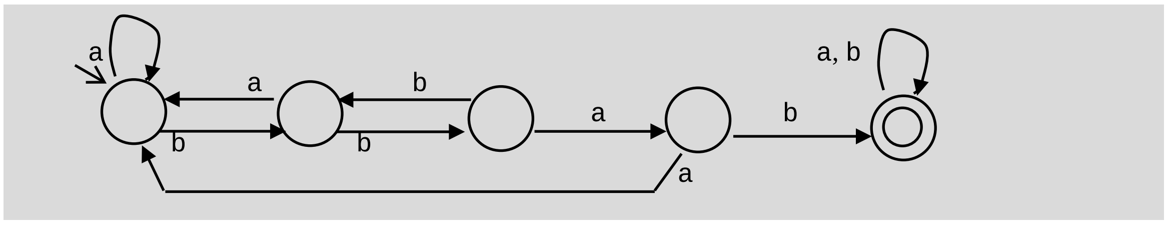
f) {*w* ∈ {0, 1}* : *w* has 001 as a substring}.



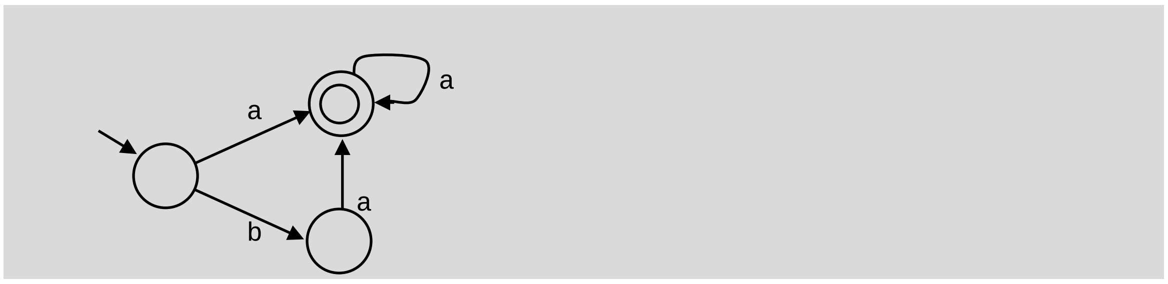
g) {*w* ∈ {0, 1}* : *w* does not have 001 as a substring}.



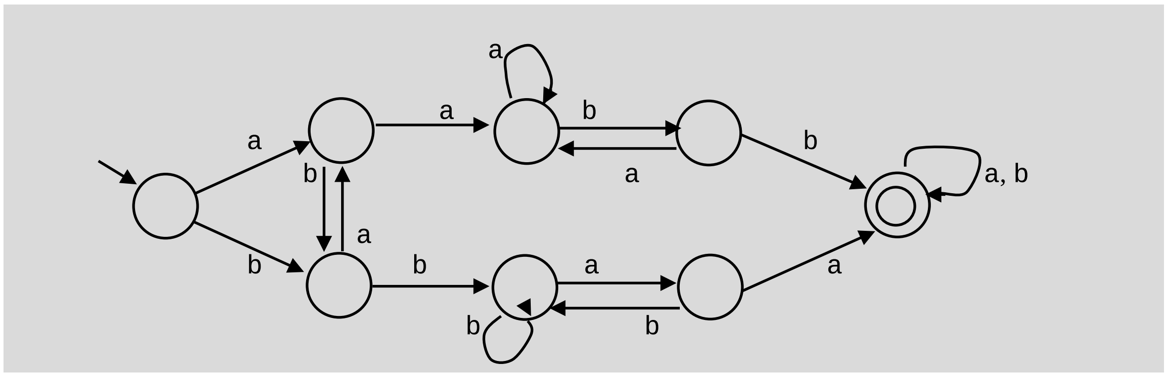
h) {*w* ∈ {a, b}* : *w* has bbab as a substring}.



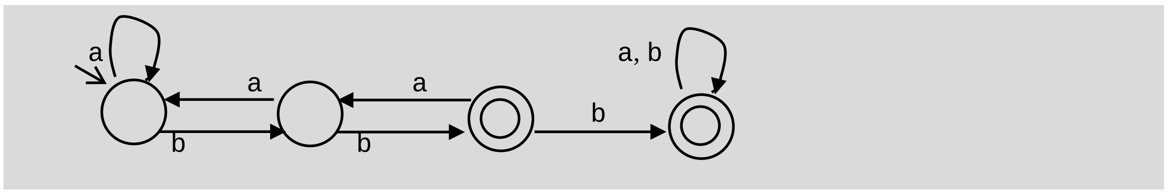
i) {*w* ∈ {a, b}* : *w* has neither ab nor bb as a substring}.



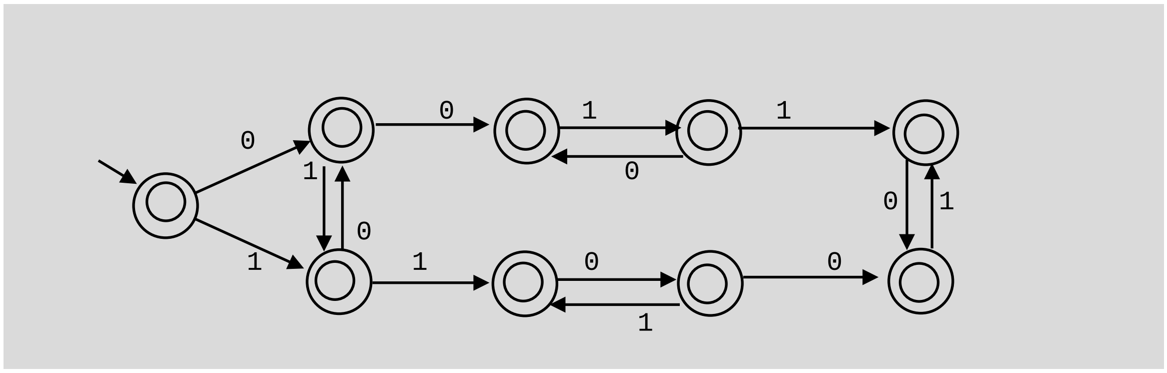
j) {*w* ∈ {a, b}* : *w* has both aa and bb as a substrings}.



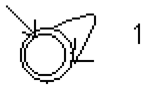
k) {*w* ∈ {a, b}* : *w* contains at least two b's that are not immediately followed by a's}.



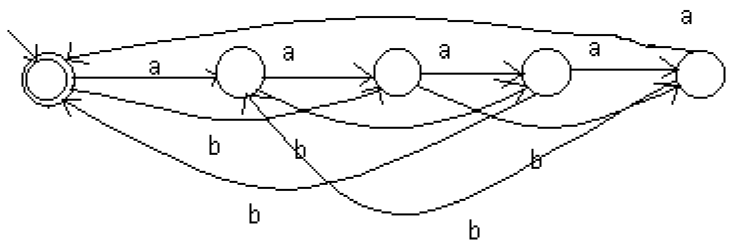
l) The set of binary strings with at most one pair of consecutive `0`'s and at most one pair of consecutive `1`'s.



m) $\{w \in \{0, 1\}^* :$ none of the prefixes of w ends in `0`$\}$.



n) $\{w \in \{a, b\}^* : (\#_a(w) + 2 \cdot \#_b(w)) \equiv_5 0\}$. ($\#_a w$ is the number of `a`'s in w).



3) Consider the children's game Rock, Paper, Scissors. We'll say that the first player to win two rounds wins the game. Call the two players *A* and *B*.

a) Define an alphabet Σ and describe a technique for encoding Rock, Paper, Scissors games as strings over Σ. (Hint: each symbol in Σ should correspond to an ordered pair that describes the simultaneous actions of *A* and *B*.)

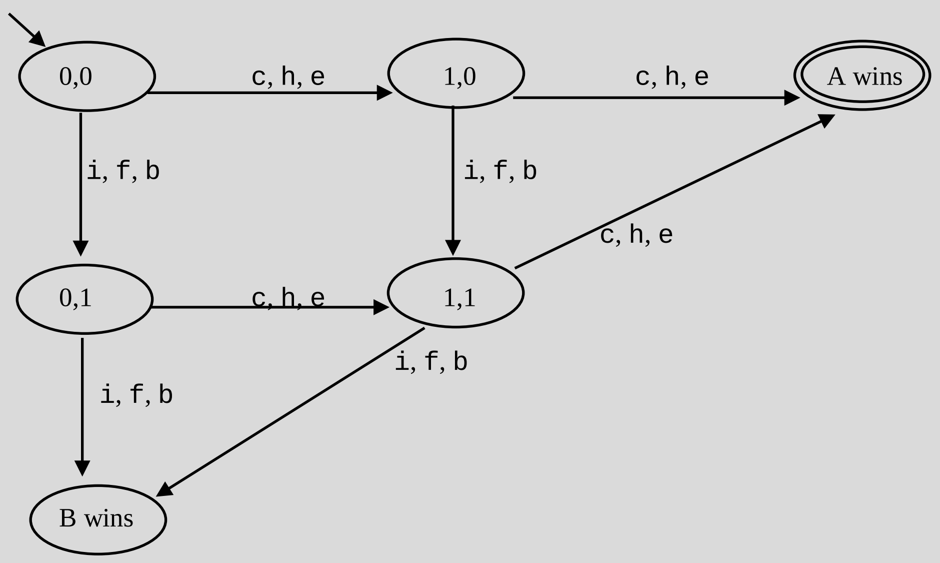
Let Σ have 9 characters. We'll use the symbols `a` - `i` to correspond to the following events. Let the first element of each pair be A's move. The second element of each pair will be B's move.

a	b	c	d	e	f	g	h	i
R, R	R, P	R, S	P, P	P, R	P, S	S, S	S, P	S, R

A Rock, Paper, Scissors game is a string of the symbols `a` - `i`. We'll allow strings of arbitrary length, but once one player as won two turns, no further events affect the outcome of the match.

b) Let L_{RPS} be the language of Rock, Paper, Scissors games, encoded as strings as described in part (a), that correspond to wins for player *A*. Show a DFSM that accepts L_{RPS}.

In the following diagram, a state with the name (*n*, *m*) corresponds to the case where player *A* has won *n* games and player *B* has one *m* games.

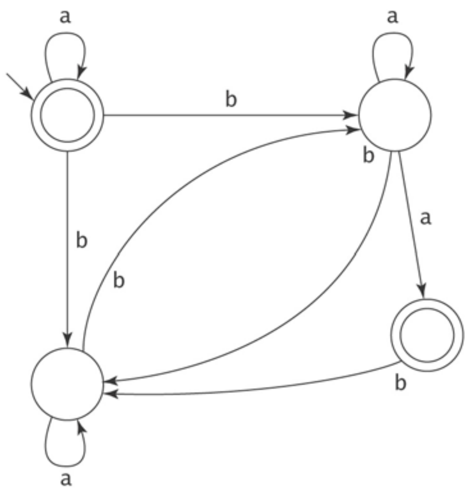


In addition, from every state, there is a transition back to itself labeled `a`, `d`, `g` (since the match status is unchanged if both players make the same move). And, from the two winning states, there is a transition back to that same state with all other labels (since, once someone has won, future events don't matter).

4) If *M* is a DFSM and ε ∈ *L*(*M*), what simple property must be true of *M*?

The start state of *M* must be an accepting state.

5) Consider the following NDFSM *M*:

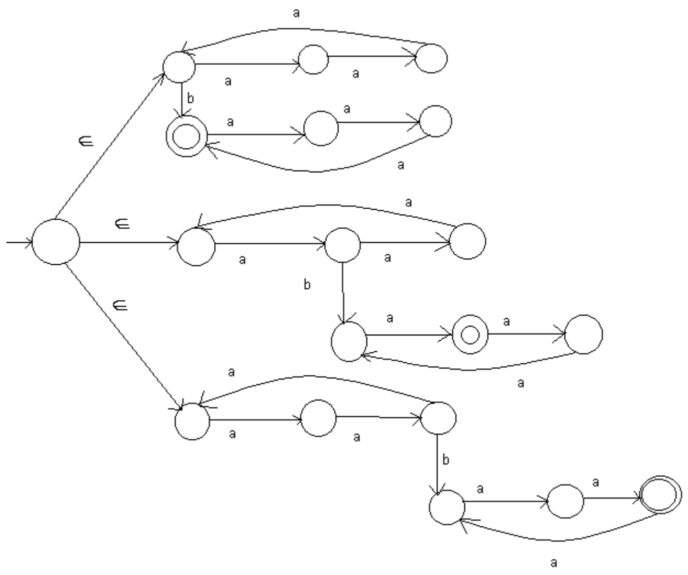


For each of the following strings *w*, determine whether *w* ∈ *L*(*M*):

a)	`aabbba`.	Yes.
b)	`bab.`	No.
c)	`baba.`	Yes.

6) Show a possibly nondeterministic FSM to accept each of the following languages:

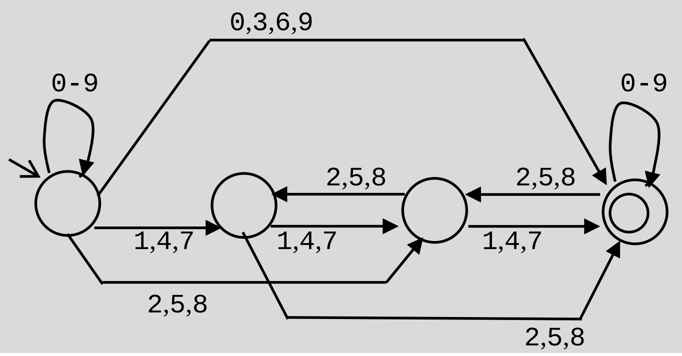
a) $\{\texttt{a}^n\texttt{b}\texttt{a}^m : n, m \geq 0, n \equiv_3 m\}$.



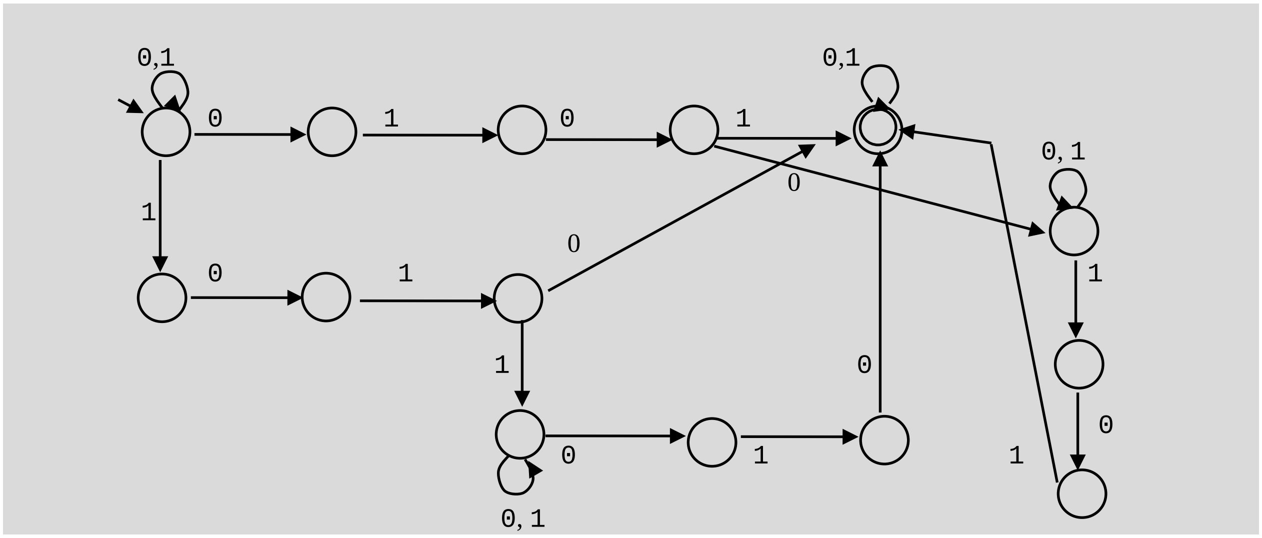
b) $\{w \in \{\texttt{a}, \texttt{b}\}^* : w$ contains at least one instance of `aaba`, `bbb` or `ababa`$\}$.

c) $L = \{w \in \{\texttt{0-9}\}^* : w$ represents the decimal encoding of a natural number whose encoding contains, as a substring, the encoding of a natural number that is divisible by 3$\}$.

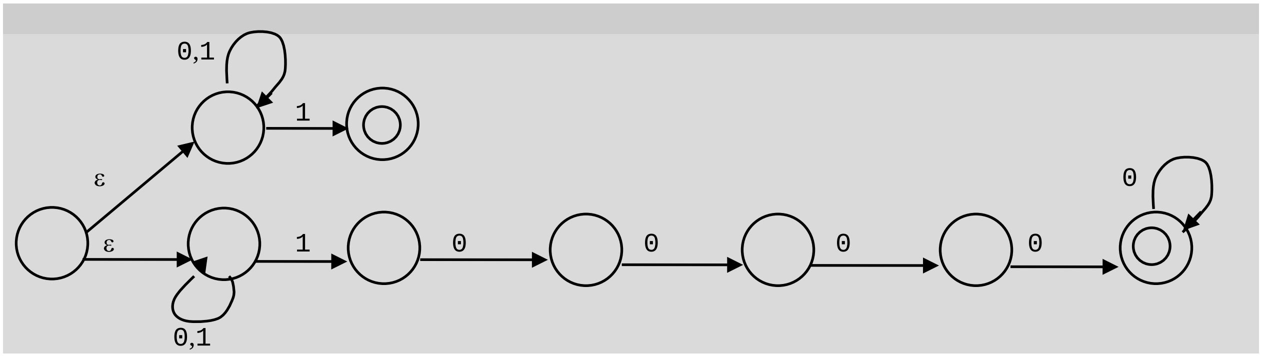
Note that 0 is a natural number that is divisible by 3. So any string that contains even one 0, 3, 6, or 9 is in *L*, no matter what else it contains. Otherwise, to be in *L*, there must be a sequence of digits whose sum equals 0 mod 3.



d) $\{w \in \{0, 1\}^* : w$ contains both `101` and `010` as substrings$\}$.



e) $\{w \in \{0, 1\}^* : w$ corresponds to the binary encoding of a positive integer that is divisible by 16 or is odd$\}$.

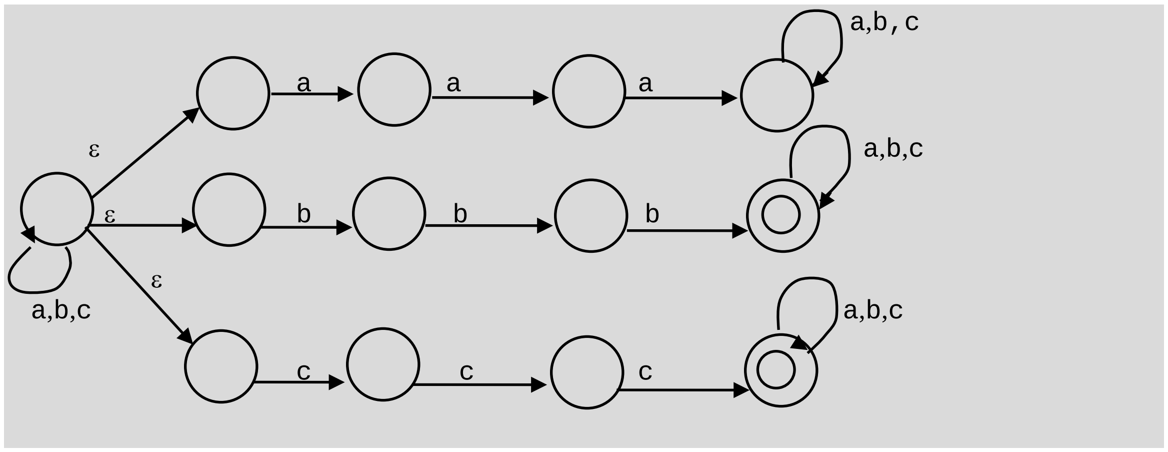


f) $\{w \in \{$a, b, c, d, e$\}^* : |w| \geq 2$ and w begins and ends with the same symbol$\}$.

Guess which of the five symbols it is. Go to a state for each. Then, from each such state, guess that the next symbol is not the last and guess that it is.

7) Show an FSM (deterministic or nondeterministic) that accepts $L = \{w \in \{\texttt{a}, \texttt{b}, \texttt{c}\}^* : w$ contains at least one substring that consists of three identical symbols in a row$\}$. For example:

- The following strings are in *L*: `aabbb`, `baacccbbb`.
- The following strings are not in *L*: ε, `aba`, `abababab`, `abcbcab`.

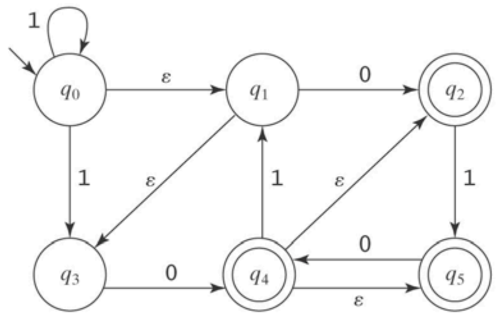


8) Show a deterministic FSM to accept each of the following languages. The point of this exercise is to see how much harder it is to build a deterministic FSM for tasks like these than it is to build an NDFSM. So do not simply built an NDFSM and then convert it. But do, after you build a DFSM, build an equivalent NDFSM.

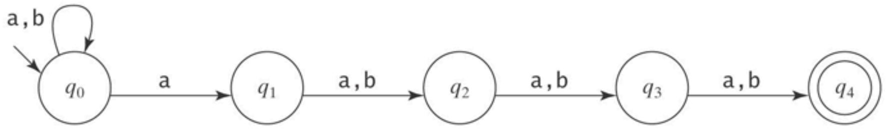
a) $\{w \in \{\texttt{a}, \texttt{b}\}^* :$ the fourth from the last character is $\texttt{a}\}$.

b) $\{w \in \{\texttt{a}, \texttt{b}\}^* : \exists x, y \in \{\texttt{a}, \texttt{b}\}^* : ((w = x\ \texttt{abbaa}\ y) \vee (w = x\ \texttt{baba}\ y))\}$.

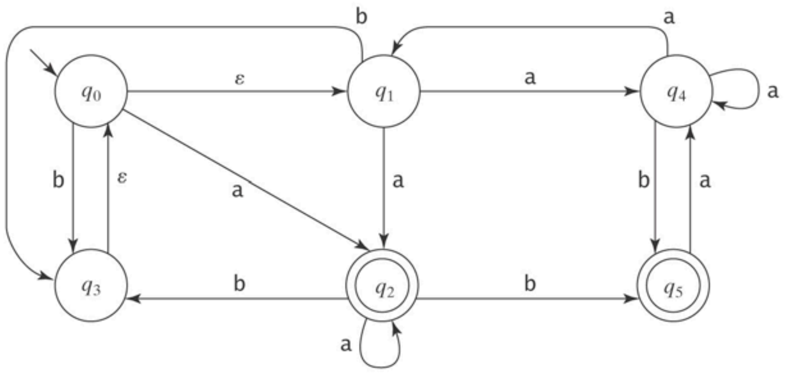
9) For each of the following NDFSMs, use *ndfsmtodfsm* to construct an equivalent DFSM. Begin by showing the value of *eps*(*q*) for each state *q*:



(a)



(b)

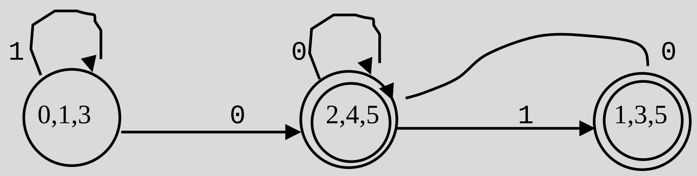


(c)

a)

s	***eps*(*s*)**
q_0	$\{q_0, q_1, q_3\}$
q_1	$\{q_1, q_3\}$
q_2	$\{q_2\}$
q_3	$\{q_3\}$
q_4	$\{q_2, q_4, q_5\}$
q_5	$\{q_5\}$

$\{q_0, q_1, q_3\}$	0	$\{q_2, q_4, q_5\}$
	1	$\{q_0, q_1, q_3\}$
$\{q_2, q_4, q_5\}$	0	$\{q_2, q_4, q_5\}$
	1	$\{q_1, q_3, q_5\}$
$\{q_1, q_3, q_5\}$	0	$\{q_2, q_4, q_5\}$
	1	{ }



b)

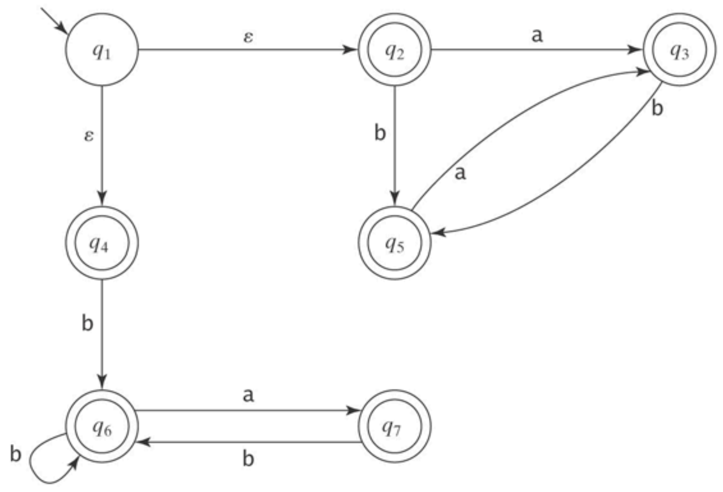
c)

s	$eps(s)$
q_0	$\{q_0, q_1\}$
q_1	$\{q_1\}$
q_2	$\{q_2\}$
q_3	$\{q_3, q_0, q_1\}$
q_4	$\{q_4\}$
q_5	$\{q_5\}$

$\{q_0, q_1\}$	a	$\{q_2, q_4\}$
	b	$\{q_0, q_1, q_3\}$
$\{q_2, q_4\}$	a	$\{q_1, q_2, q_4\}$
	b	$\{q_0, q_1, q_3, q_5\}$
$\{q_0, q_1, q_3\}$	a	$\{q_2, q_4,\}$
	b	$\{q_0, q_1, q_3\}$
$\{q_1, q_2, q_4\}$	a	$\{q_1, q_2, q_4\}$
	b	$\{q_0, q_1, q_3, q_5\}$
$\{q_0, q_1, q_3, q_5\}$	a	$\{q_2, q_4\}$
	b	$\{q_0, q_1, q_3\}$

Accepting state is $\{q_0, q_1, q_3, q_5\}$. 1

10) Let M be the following NDFSM. Construct (using *ndfsmtodfsm*), a DFSM that accepts $\neg L(M)$.



1) Complete by creating Dead state D and adding the transitions: {3, a, D}, {5, b, D}, {4, a, D}, {7, a, D}, {D, a, D}, {D, b, D}.

2) Convert to deterministic:

eps{1} = {1, 2, 4}

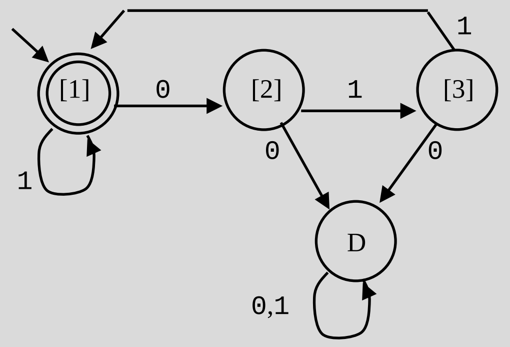
{1, 2, 4}, a, {3, *D*}
{1, 2, 4}, b, {5, 6}
{3, *D*}, a, {*D*}
{3, *D*}, b {5, *D*}
{5, 6}, a, {3, 7}
{5, 6}, b, {*D*, 6}
{*D*}, a, {*D*}
{*D*}, b, {*D*}
{5, *D*}, a, {3, *D*}
{5, *D*}, b, {*D*}
{3, 7}, a, {*D*}
{3, 7}, b, {5, 6}
{*D*, 6}, a, {*D*, 7}
{*D*, 6}, b, {*D*, 6}
{*D*, 7}, a , {*D*}
{*D*, 7}, b, {*D*, 6}
All these states are accepting except {*D*}.

3) Swap accepting and nonnonaccepting states, making all states nonaccepting except {*D*).

11) For each of the following languages *L*:
  (i) Describe the equivalence classes of $\approx_L$.
  (ii) If the number of equivalence classes of $\approx_L$ is finite, construct the minimal DFSM that accepts *L*.

a) {*w* ∈ {0, 1}* : every 0 in *w* is immediately followed by the string 11}.

[1] {in *L*}
[2] {otherwise in *L* except ends in 0}
[3] {otherwise in *L* except ends in 01}
[D] {corresponds to the Dead state: string contains at least one instance of 00 or 010}



b) {*w* ∈ {0, 1}* : *w* has either an odd number of 1's and an odd number of 0's or it has an even number of 1's and an even number of 0's}.

c) {*w* ∈ {a, b}* : *w* contains at least one occurrence of the string aababa}.

d) $\{ww^R : w \in \{\texttt{a}, \texttt{b}\}^*\}$.

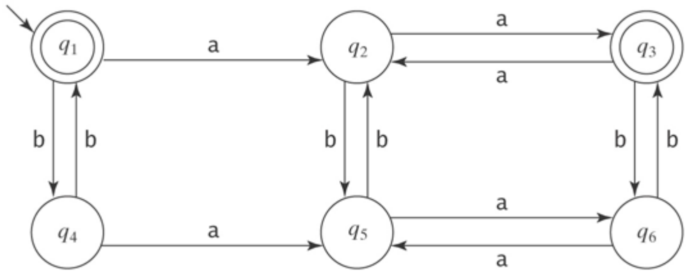
[1] {ε}	in *L*
[2] {a}	
[3] {b}	
[4] {aa}	in *L*
[5] {ab}	

And so forth. Every string is in a different equivalence class because each could become in *L* if followed by the reverse of itself but not if followed by most other strings. This language is not regular.

e) $\{w \in \{\texttt{a}, \texttt{b}\}^* :$ *w* contains at least one a and ends in at least two b's}.

f) $\{w \in \{\texttt{0}, \texttt{1}\}^* :$ there is no occurrence of the substring 000 in *w*}.

12) Let *M* be the following DFSM. Use *minDFSM* to minimize *M*.



Initially, *classes* = {[1, 3], [2, 4, 5, 6]}.

At step 1:

((1, a), [2, 4, 5, 6])	((3, a), [2, 4, 5, 6])		No splitting required here.
((1, b), [2, 4, 5, 6])	((3, b), [2, 4, 5, 6])		
((2, a), [1, 3])	((4, a), [2, 4, 5, 6])	((5, a), [2, 4, 5, 6])	((6, a), [2, 4, 5, 6])
((2, b), [2, 4, 5, 6])	((4, b), [1, 3])	((5, b), [2, 4, 5, 6])	((6, b), [1, 3])

These split into three groups: [2], [4, 6], and [5]. So classes is now {[1, 3], [2], [4, 6], [5]}.

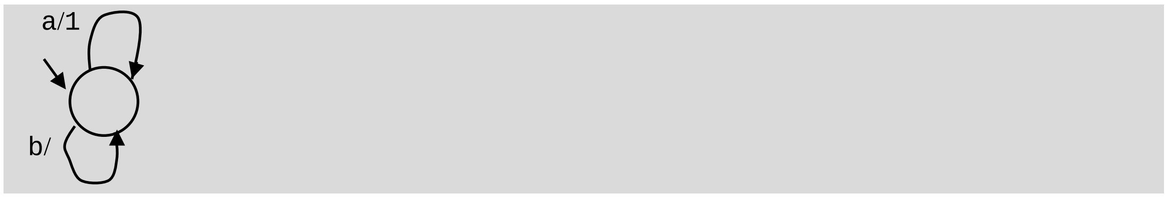
At step 2, we must consider [4, 6]:

((4, a), [5])	((6, a), [5])
((4, b), [1])	((6, b), [1])

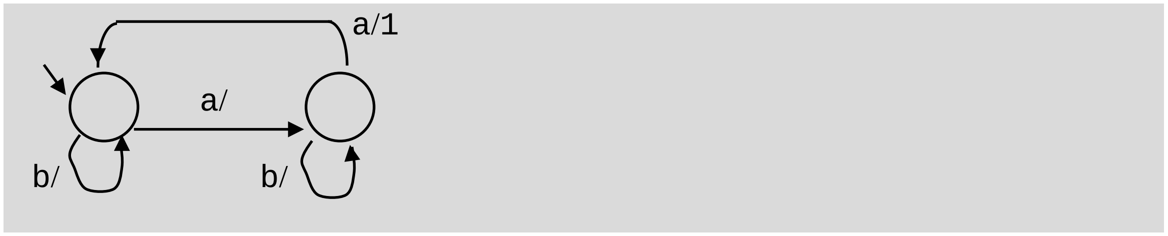
No further splitting is required. The minimal machine has the states: {[1, 3], [2], [4, 6], [5]}, with transitions as shown above.

13) Construct a deterministic finite state transducer with input {a, b} for each of the following tasks:

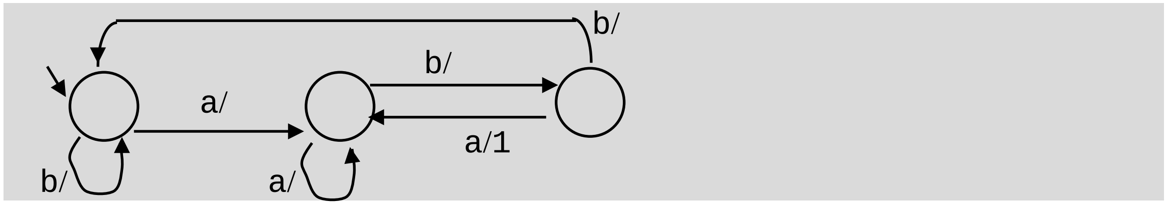
a) On input *w*, produce 1^n, where $n = \#_a(w)$.



b) On input *w*, produce 1^n, where $n = \#_a(w)/2$.



c) On input *w*, produce 1^n, where *n* is the number of occurrences of the substring aba in *w*.



14) Construct a deterministic finite state transducer that could serve as the controller for an elevator. Clearly describe the input and output alphabets, as well as the states and the transitions between them.

15) Consider the problem of counting the number of words in a text file that may contain letters plus any of the following characters:

<blank> *<linefeed>* *<end-of-file>* , . ; : ? !

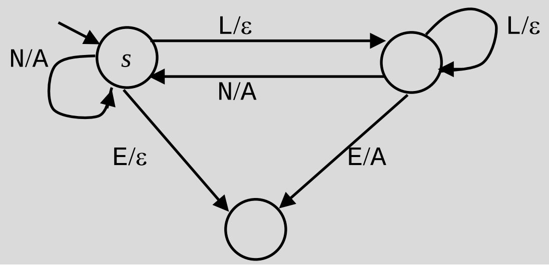
Define a word to be a string of letters that is preceded by either the beginning of the file or some non-letter character and that is followed by some non-letter character. For example, there are 11 words in the following text:

The *<blank>* *<blank>* cat *<blank>* *<linefeed>*
saw *<blank>* the *<blank>* *<blank>* *<blank>* rat *<linefeed>*
<blank> with
<linefeed> a *<blank>* hat *<linefeed>*
on *<blank>* the *<blank>* *<blank>* mat *<end-of-file>*

Describe a very simple finite-state transducer that reads the characters in the file one at a time and solves the word-counting problem. Assume that there exists an output symbol with the property that, every time it is generated, an external counter gets incremented.

We'll let the input alphabet include the symbols: L (for a letter), N (for a nonletter), and E (for end-of-file). The output alphabet will contain just a single symbol A (for add one to the counter). We don't need any accepting states.

Let $M = \{K, \{\text{L}, \text{N}, \text{E}\}, \{\text{A}\}, s, \varnothing, \delta, D\}$, where K, δ, and D are as follows.



16) Real traffic light controllers are more complex than the one that we drew in Example 5.29.
   a) Consider an intersection of two roads controlled by a set of four lights (one in each direction). Don't worry about allowing for a special left-turn signal. Design a controller for this four-light system.
   b) As an emergency vehicle approaches an intersection, it should be able to send a signal that will cause the light in its direction to turn green and the light in the cross direction to turn yellow and then red. Modify your design to allow this.

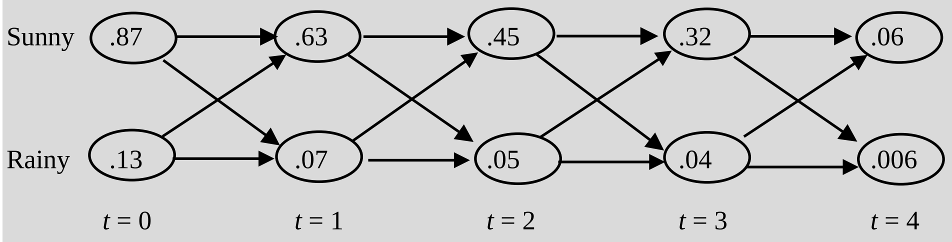
17) Real bar code systems are more complex than the one we sketched in the book. They must be able to encode all ten digits, for example. There are several industry-standard formats for bar codes, including the common UPC code found on nearly everything we buy. Search the web. Find the encoding scheme used for UPC codes. Describe a finite state transducer that reads the bars and outputs the corresponding decimal number.

18) Extend the description of the Soundex FSM that was started in Example 5.33 so that it can assign a code to the name Pfifer. Remember that you must take into account the fact that every Soundex code is made up of exactly four characters.

19) Consider the weather/passport HMM of Example 5.37. Trace the execution of the Viterbi and forward algorithms to answer the following questions:

Students can solve these problems by hand, by writing code, or by using standard packages. These solutions were created using the HMM package in Matlab.

   a) Suppose that the report ###L is received from Athens. What was the most likely weather during the time of the report?

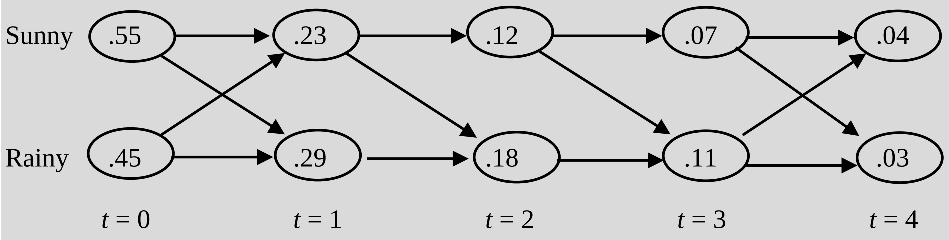


So the most likely series of weather reports is Sunny, Sunny, Sunny, Sunny.

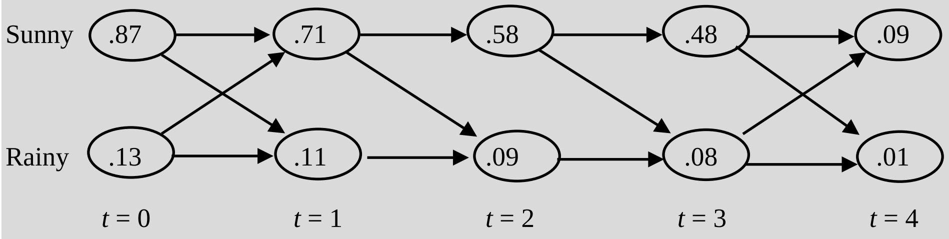
b) Is it more likely that ###L came from London or from Athens?

To solve this problem, we run the forward algorithm on both the London and the Athens model and see which has the higher probability of outputting the observed sequence. So we have:

London:



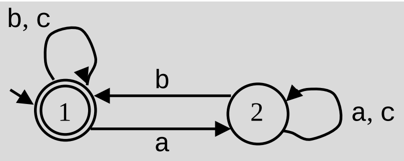
Athens:



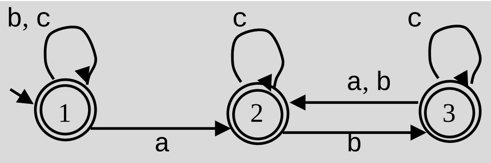
The total probability for London is thus .07. The total probability for Athens is .1. So Athens is more likely to have produced the output ###L.

20) Construct a Büchi automaton to accept each of the following languages of infinite length strings:

a) $\{w \in \{a, b, c\}^\omega$: after any occurrence of an a there is eventually an occurrence of a b$\}$.

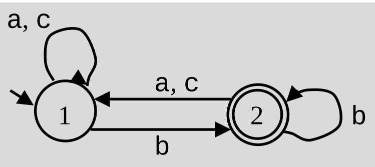


b) $\{w \in \{a, b, c\}^\omega$: between any two a's there is an odd number of b's$\}$.



We have omitted the dead state, which is reached from state 2 on an a.

c) $\{w \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^{\omega}$: there never comes a time after which no b's occur$\}$.



21) In H.2, we describe the use of statecharts as a tool for building complex systems. A statechart is a hierarchically structured transition network model. Statecharts aren't the only tools that exploit this idea. Another is Simulink® 💻, which is one component of the larger programming environment Matlab® 💻. Use Simulink to build an FSM simulator.

22) In I.1.2, we describe the Alternating Bit Protocol for handling message transmission in a network. Use the FSM that describes the sender to answer the question, "Is there any upper bound on the number of times a message may be retransmitted?"

No. The loop from either of the need ACK states to the corresponding timeout state can happen any number of times.

23) In J.1, we show an FSM model of simple intrusion detection device that could be part of a building security system. Extend the model to allow the system to have two zones that can be armed and disarmed independently of each other.